UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/600,256	06/19/2003	Brian Keith Pepin	13768-406	3628

47973 7590 06/21/2007

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

MAIL DATE	DELIVERY MODE
06/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/600,256	PEPIN ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 May 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7,9-16,18-26,28-33,35-42,45 and 46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7,9-16,18-26,28-33,35-42,45 and 46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *3/26/07, 4/13/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on May 14, 2007 has been entered. Claims 1-7, 9-16, 18-26, 28-33, 35-42, 45 and 46 are pending.

Response to Arguments

2. Applicant's arguments have been fully considered but they are not persuasive.

As noted in the final Office action mailed on March 16, 2007, Barnes teaches producing a snippet of code in the form of XML (see, for example, column 4, lines 30-35 and 44-46). The examiner also noted that the XML representation of the object graph does not constitute a class representation of the object graph. Thus, Barnes teaches producing a snippet of code in the form of XML without producing a class representation of the object graph.

Applicant contends that a conclusion that an XML document is not a class representation is without support in any of the cited references (remarks, page 14). Applicant further contends that even if such a conclusion is accepted, and an XML representation is not itself a class representation, "the cited references fail to have any disclosure that ... there are no class representations produced within the XML representation" (remarks, page 14). In other words, Applicant argues that if the XML representation includes a class representation of the object

graph, it does not teach generating a snippet of code "without ... a class representation of the object graph" as recited in the claims (remarks, page 14).

However, the examiner respectfully submits that Applicant's arguments are not commensurate with the scope of the claims. The specification does not provide an explicit and deliberate definition of the term "class representation." Rather, Applicant's specification states that "conventional serialization ... typically produces a new class, like the new About class shown in Table I" (page 3, paragraph [0007]), and illustrates in Table I (page 2) what one of ordinary skill in the art would consider a "class representation." Thus, a reasonable interpretation of the claims is merely that the serializer produces a snippet of code without producing a representation in a form such as shown in Table I.

Indeed, the XML representation produced in Barnes is not what one of ordinary skill in the art would consider a "class representation," and clearly does not include code in the same form as what Applicant provides in Table I. Barnes illustrates the XML code in FIGS. 6A-6C, and teaches that the XML code includes tag elements (see, for example, column 8, lines 43-61) rather than any class representation.

While the examiner agrees with Applicant's conclusion that an XML representation does not inherently *not* include a class representation (remarks, page 15), the XML file described in U.S. Patent No. 7,020,641 to Leong et al. is in no way representative of Barnes. The XML code specifically described in Barnes does not include a "class representation" within the meaning that Applicant's specification implies.

Accordingly, the examiner respectfully submits that Barnes in view of Chinnici and Ardoin teaches or suggests that "the serializer produces a snippet of code sufficient to undo or

redo a change to the object graph, but without producing a class representation of the object graph," such as recited in the claims.

Moreover, the examiner notes that the language "without producing a class representation of the object graph" amounts to a negative limitation, and proposes that Applicant instead particularly point out and distinctly claim what *is* produced, rather than attempting to exclude what is not produced. See MPEP § 2173.05(i).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-7, 9-16, 18-26, 28-33, 35-42, 45 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 7,096,419 to Barnes et al. (art of record, "Barnes") in view of U.S. Pub. No. 2003/0191803 to Chinnici et al. (art of record, "Chinnici") and in view of U.S. Patent No. 6,052,691 to Ardoin et al. (art of record, "Ardoin").

With respect to claim 1 (currently amended), Barnes discloses, in a computer system that supports a serialization engine capable of generating source code for objects (see, for example, column 2, lines 46-54, which shows a serialization engine and code generator), a method of serializing one or more objects from an initial representation to any of one or more subsequent representations (see, for example, column 3, lines 32-34 and 51-52, which shows serializing

objects from an initial representation to a subsequent representation), for one or more standard object types and serialization formats (see, for example, column 5, lines 21-25, which show shows standard object types, and column 3, lines 55-62, which shows standard serialization formats), and which may be extended to cover one or more custom object types and serialization formats (see, for example, column 5, lines 47-53, which shows custom object types, and column 4, lines 10-14, which shows custom serialization formats).

Barnes does not expressly disclose acts of:

providing a serialization manager to (i) coordinate one or more standard serialization providers that each identify one or more standard serializers for a standard object type or serialization format, and (ii) load, as needed, one or more custom serialization providers that each identify one or more custom serializers for one or more custom object types or serialization formats that are not covered by the one or more standard serialization providers.

However, in an analogous art, Chinnici discloses a pluggable serialization engine (see, for example, paragraph 0126, lines 1-13) that supports one or more object types (see, for example, paragraph 0129, lines 1-9) and serialization formats (see, for example, paragraph 0127, line 1-8). Chinnici further discloses providing a serialization manager to coordinate and load the needed serialization providers (see, for example, paragraph 0140, lines 1-11). One advantage of the pluggable serialization engine is that the serializers are independently developed and support flexible mappings among object types and serialization formats (see, for example, paragraph 0139, lines 1-20).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes with a pluggable serialization engine such as

taught in Chinnici, so as to enable the use of independently developed serializers that support flexible mappings among object types and serialization formats.

Barnes in view of Chinnici further discloses acts of:

requesting a serializer from the serialization manager for an object graph that comprises an object of a particular object type, and for a particular serialization format (see, for example, Chinnici, paragraph 0141, lines 1-5, which shows requesting a serializer for a particular serialization format, and paragraph 0152, lines 8-10, which further shows that the serializer is requested for a particular object type); and

calling the serializer to serialize the object graph (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows serializing the object graph).

Barnes in view of Chinnici further discloses that the serializer produces a snippet of code without producing a class representation of the object graph (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows producing a snippet of code in the form of XML rather than a class representation of the object graph, and FIGS. 6A-6C, which further shows XML code that does not include a class representation of the object graph), but does not expressly disclose that the serializer is requested from the serialization manager as part of a cut, copy or paste operation and such that due to the cut, copy or paste operation, the serializer produces a snippet of code sufficient to undo or redo a change to the object graph.

However, in an analogous art, Ardoin discloses a system for modeling an object graph (see, for example, column 6, lines 28-39) that preserves data and referential integrity as part of copy and delete operations, and enables one to undo any such changes to the object graph (see,

for example, column 3, lines 13-29) made in a visual user interface designer (see, for example, column 1, lines 58 to column 2, line 2).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes and Chinnici such that the serializer is requested from the serialization manager as part of a cut, copy or paste operation, and such that due to the cut, copy or paste operation, the serializer produces a snippet of code sufficient to undo or redo a change to the object graph, as Ardoin suggests, so as provide such operations wherein data and referential integrity are preserved.

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serializer is a custom serializer (see, for example, Chinnici, paragraph 0128, lines 14-29).

With respect to claim 3 (original), the rejection of claim 2 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serialization manager automatically loads the one or more custom serialization providers in response to the request (see, for example, Chinnici, paragraph 0153, lines 1-5, which shows loading the custom serializer).

With respect to claim 4 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the computer system also supports a visual user interface designer and the serialization engine is capable of generating source code for one or more user interface objects created within the visual user interface designer (see, for example, Barnes, column 3, lines 23-31, which shows a visual user interface designer, and column 4, lines 60-65, which shows generating source code for the user interface objects), the

object graph comprising the one or more user interface objects (see, for example, Barnes, column 6, lines 4-33, which shows that the object graph comprises the user interface objects).

With respect to claim 5 (previously presented), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serialization manager maintains context information that can be shared among the one or more standard serialization providers and any custom serialization providers that are loaded by the serialization manager (see, for example, Chinnici, paragraph 0143, lines 1-12, which shows maintaining context information).

With respect to claim 6 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the initial representation comprises a representation used to persist the one or more user interface objects, and wherein the one or more subsequent representations comprise a representation used to represent the one or more user interface objects within the visual user interface designer (see, for example, Barnes, column 4, lines 47-59, which shows an initial representation for persisting the user interface objects and a subsequent representation for representing the user interface objects within the visual user interface designer).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the object graph comprises a plurality of related objects (see, for example, Barnes, column 6, lines 4-33, which shows that the object graph comprises a plurality of related objects).

With respect to claim 9 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the one or more standard serialization providers are capable of identifying a plurality of serializers (see, for example, Chinnici, paragraph 0141, lines 7-9, which shows identifying a plurality of serializers), and wherein at least two of the plurality of serializers are for serializing the object graph in different serialization formats (see, for example, Chinnici, paragraph 0128, lines 14-29, which shows a plurality of serializers for different serialization formats).

With respect to claim 10 (original), the rejection of claim 1 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serializer produces an eXtensible Markup Language representation of the object graph (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows producing an XML representation of the object graph).

With respect to claim 11 (currently amended), the claim is directed to a computer program product that corresponds to the method of claim 1 (see the rejection of claim 1 above). Barnes in view of Chinnici in view of Ardoin further discloses that the computer system supports a visual user interface designer and a serialization engine capable of generating source code for user interface objects created within the visual user interface designer (see the rejection of claim 4 above).

With respect to claim 12 (original), the rejection of claim 11 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the one or more custom object types or serialization formats are not covered by the one or more standard serialization providers (see,

for example, Chinnici, paragraph 0134, lines 1-14, which shows that serializers for custom object types or serialization formats may override the default serialization providers).

With respect to claim 13 (original), the rejection of claim 11 is incorporated, and the claim corresponds to claim 2 (see the rejection of claim 2 above).

With respect to claim 14 (original), the rejection of claim 13 is incorporated, and the claim corresponds to claim 3 (see the rejection of claim 3 above).

With respect to claim 15 (currently amended), the rejection of claim 11 is incorporated, and the claim corresponds to claim 5 (see the rejection of claim 5 above).

With respect to claim 16 (original), the rejection of claim 11 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the initial representation comprises a live representation used to represent the one or more user interface objects within the visual user interface designer, and wherein the one or more subsequent representations comprise a target representation used to persist the one or more user interface objects (see, for example, Barnes, column 2, lines 34-45, which shows an initial representation for representing the user interface objects within the visual user interface designer and a subsequent representation for persisting the user interface objects).

With respect to claim 18 (original), the rejection of claim 11 is incorporated, and the claim corresponds to claim 9 (see the rejection of claim 9 above).

With respect to claim 19 (original), the rejection of claim 11 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serializer produces a source code representation of the object graph (see, for example, Barnes, column 4, lines 60-65, which shows producing a source code representation of the object graph).

With respect to claim 20 (currently amended), Barnes discloses, in a computer system that supports a visual user interface designer and a serialization engine capable of generating source code for user interface objects created within the visual user interface designer (see, for example, column 2, lines 46-54, which shows a serialization engine and code generator, and column 3, lines 23-31, which further shows a visual user interface designer), a method of serializing one or more user interface objects from an initial representation to any of one or more extensible subsequent representations which may be extended to cover one or more custom object types and serialization formats (see, for example, column 3, lines 32-34 and 51-52, which shows serializing objects from an initial representation to a subsequent representation, and see, for example, column 5, lines 47-53, which shows custom object types, and column 4, lines 10-14, which shows custom serialization formats).

Barnes does not expressly disclose steps for:

coordinating one or more standard serialization providers that each identify one or more standard serializers for a standard object type or serialization format; and

loading, as needed, one or more custom serialization providers that each identify one or more custom serializers for one or more custom object types or serialization formats that are not covered by the one or more standard serialization providers.

However, in an analogous art, Chinnici discloses a pluggable serialization engine (see, for example, paragraph 0126, lines 1-13) that supports one or more object types (see, for example, paragraph 0129, lines 1-9) and serialization formats (see, for example, paragraph 0127, line 1-8). Chinnici further discloses providing a serialization manager to coordinate and load the needed serialization providers (see, for example, paragraph 0140, lines 1-11). One advantage of the pluggable serialization engine is that the serializers are independently developed and support flexible mappings among object types and serialization formats (see, for example, paragraph 0139, lines 1-20).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes with a pluggable serialization engine such as taught in Chinnici, so as to enable the use of independently developed serializers that support flexible mappings among object types and serialization formats.

Barnes in view of Chinnici further discloses steps for:

identifying a serializer for a particular serialization format and for an object graph that comprises an object of a particular object type (see, for example, Chinnici, paragraph 0141, lines 1-5, which shows identifying a serializer for a particular serialization format, and paragraph 0152, lines 8-10, which further shows that the serializer is identified for a particular object type); and

serializing the object graph with the identified serializer (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows serializing the object graph).

Barnes in view of Chinnici further discloses that the serializer produces a snippet of code without producing a class representation of the object graph (see, for example, Barnes, column 4,

lines 30-35 and 44-46, which shows producing a snippet of code in the form of XML rather than a class representation of the object graph, and FIGS. 6A-6C, which further shows XML code that does not include a class representation of the object graph), but does not expressly disclose that serializing the object graph with the identified serializer is performed as part of a cut, copy or paste operation, such that due to the cut, copy or paste operation, the serializer produces a snippet of code sufficient to undo or redo a change to the object graph made within the visual user interface designer.

However, in an analogous art, Ardoin discloses a system for modeling an object graph (see, for example, column 6, lines 28-39) that preserves data and referential integrity as part of copy and delete operations, and enables one to undo any such changes to the object graph (see, for example, column 3, lines 13-29) made in a visual user interface designer (see, for example, column 1, lines 58 to column 2, line 2).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes and Chinnici such that serializing the object graph with the identified serializer is performed as part of a cut, copy or paste operation, and such that due to the cut, copy or paste operation, the serializer produces a snippet of code sufficient to undo or redo a change to the object graph made within the visual user interface designer, as Ardoin suggests, so as provide such operations wherein data and referential integrity are preserved.

With respect to claim 21 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the particular object type comprises a

custom object type (see, for example, Barnes, column 5, lines 47-53, which shows custom object types).

With respect to claim 22 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the particular serialization format is a custom serialization format (see, for example, Barnes, column 4, lines 10-14, which shows custom serialization formats).

With respect to claim 23 (original), the rejection of claim 23 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses a step for maintaining context information to be shared among the one or more standard serialization providers and any custom serialization providers that are loaded by the serialization manager (see, for example, Chinnici, paragraph 0143, lines 1-12, which shows maintaining context information).

With respect to claim 24 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the initial representation comprises a live representation used to represent the one or more user interface objects within the visual user interface designer, and wherein the one or more subsequent representations comprise a target representation used to persist the one or more user interface objects (see, for example, Barnes, column 2, lines 34-45, which shows an initial representation for representing the user interface objects within the visual user interface designer and a subsequent representation for persisting the user interface objects).

With respect to claim 25 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses a step for replacing a standard serializer with a custom serializer from a custom serialization provider loaded while identifying the serializer for the object graph (see, for example, Chinnici, paragraph 0134, lines 1-14, which shows that serializers for custom object types or serialization formats may override the default serialization providers).

With respect to claim 26 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the object graph comprises a plurality of related objects (see, for example, Barnes, column 6, lines 4-33, which shows that the object graph comprises a plurality of related objects).

With respect to claim 28 (original), the rejection of claim 20 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serializer produces either a source code representation or an eXtensible Markup Language representation of the object graph (see, for example, Barnes, column 4, lines 60-65, which shows producing a source code representation of the object graph, and column 4, lines 30-35 and 44-46, which shows producing an XML representation of the object graph).

With respect to claim 29 (currently amended), the claim is directed to a computer program product that corresponds to the method of claim 20 (see the rejection of claim 20 above).

With respect to claim 30 (original), the rejection of claim 29 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that one or more custom serialization providers are loaded in identifying the serializer for the object graph (see, for example, Chinnici, paragraph 0153, lines 1-5, which shows loading the custom serializer).

With respect to claim 31 (original), the rejection of claim 29 is incorporated, and the claim corresponds to claim 23 (see the rejection of claim 23 above).

With respect to claim 32 (original), the rejection of claim 29 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the initial representation comprises a representation used to persist the one or more user interface objects, and wherein the one or more subsequent representations comprise a representation used to represent the one or more user interface objects within the visual user interface designer (see, for example, Barnes, column 4, lines 47-59, which shows an initial representation for persisting the user interface objects and a subsequent representation for representing the user interface objects within the visual user interface designer).

With respect to claim 33 (original), the rejection of claim 29 is incorporated, and the claim corresponds to claim 25 (see the rejection of claim 25 above).

With respect to claim 35 (original), the rejection of claim 29 is incorporated, and the claim corresponds to claim 28 (see the rejection of claim 28 above).

With respect to claim 36 (previously presented), Barnes discloses, for a computer system that supports a visual user interface designer and a serialization engine capable of generating

source code for user interface objects created within the visual user interface designer (see, for example, column 2, lines 46-54, which shows a serialization engine and code generator, and column 3, lines 23-31, which further shows a visual user interface designer), a computer program product comprising one or more computer readable storage media having stored thereon computer executable instructions that implement a method of efficiently serializing one or more user interface objects when performing visual operations on the one or more user interface objects (see, for example, column 4, lines 15-29 and 47-59, which shows serializing user interface objects when performing visual operations).

Barnes does not expressly disclose steps for:

loading one or more serialization providers that each identify one or more serializers for a given object type and serialization format.

However, in an analogous art, Chinnici discloses a pluggable serialization engine (see, for example, paragraph 0126, lines 1-13) that supports one or more object types (see, for example, paragraph 0129, lines 1-9) and serialization formats (see, for example, paragraph 0127, line 1-8). Chinnici further discloses providing a serialization manager to coordinate and load the needed serialization providers (see, for example, paragraph 0140, lines 1-11). One advantage of the pluggable serialization engine is that the serializers are independently developed and support flexible mappings among object types and serialization formats (see, for example, paragraph 0139, lines 1-20).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes with a pluggable serialization engine such as

taught in Chinnici, so as to enable the use of independently developed serializers that support flexible mappings among object types and serialization formats.

Barnes in view of Chinnici further discloses steps for:

from the one or more serialization providers, identifying a serializer capable of serializing a user interface object in a serialization format that corresponds to a requested visual operation to be performed on the user interface object (see, for example, Chinnici, paragraph 0141, lines 1-5, which shows identifying a serializer for the desired serialization format); and

serializing the user interface object with the identified serializer (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows serializing the user interface object).

Barnes in view of Chinnici further discloses that the serializer produces a snippet of code without producing a class representation of the user interface object (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows producing a snippet of code in the form of XML rather than a class representation of the user interface object, and FIGS. 6A-6C, which further shows XML code that does not include a class representation of the user interface object), but does not expressly disclose that serializing the user interface object with the identified serializer is performed as part of a cut, copy, paste, undo or redo operation, such that due to the cut, copy, paste, undo or redo operation, the serializer produces a snippet of code sufficient to undo or redo a change to the user interface object made within the visual user interface designer without producing a class representation of the user interface object in order to efficiently perform the requested operation.

However, in an analogous art, Ardoin discloses a system for modeling an object graph (see, for example, column 6, lines 28-39) that preserves data and referential integrity as part of

copy and delete operations, and enables one to perform undo operations to undo any such changes to the object graph (see, for example, column 3, lines 13-29) made in a visual user interface designer (see, for example, column 1, lines 58 to column 2, line 2).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the system of Barnes and Chinnici such that serializing the user interface object with the identified serializer is performed as part of a cut, copy, paste, undo or redo operation, and such that due to the cut, copy, paste, undo or redo operation, the serializer produces a snippet of code sufficient to undo or redo a change to the user interface object made within the visual user interface designer without producing a class representation of the user interface object in order to efficiently perform the requested operation, as Ardoin suggests, so as provide such operations wherein data and referential integrity are preserved.

With respect to claim 37 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses a step for coordinating the one or more serialization providers (see, for example, Chinnici, paragraph 0140, lines 1-11, which shows coordinating the serializers).

With respect to claim 38 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the one or more serialization providers comprise one or more standard serialization providers that each identify one or more standard serializers for a standard object type and serialization format (see, for example, Chinnici, paragraph 0141, lines 7-9, which shows identifying a plurality of serializers, and

paragraph 0129, lines 1-9, which shows serializers for standard object types and serialization formats).

With respect to claim 39 (original), the rejection of claim 38 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the one or more serialization providers comprise one or more custom serialization providers that each identify one or more custom serializers for one or more custom object types or serialization formats that are not covered by the one or more standard serialization providers (see, for example, Chinnici, paragraph 0141, lines 7-9, which shows identifying a plurality of serializers, and paragraph 0128, lines 14-29, which shows serializers for custom object types and serialization formats).

With respect to claim 40 (original), the rejection of claim 39 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that one or more custom serialization providers are loaded in identifying the serializer (see, for example, Chinnici, paragraph 0153, lines 1-5, which shows loading the custom serializer).

With respect to claim 41 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses that the serializer is a custom serializer (see, for example, Chinnici, paragraph 0128, lines 14-29).

With respect to claim 42 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses a step for maintaining context information to be shared among the one or more serialization providers (see, for example, Chinnici, paragraph 0143, lines 1-12, which shows maintaining context information).

With respect to claim 45 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses steps for:

from the one or more serialization providers, identifying a code serializer capable of serializing a user interface object in a serialization format that corresponds to a source code representation of the user interface object (see, for example, Barnes, column 4, lines 60-65, which show identifying a serializer for serializing a user interface objects in a source code representation); and

serializing the user interface object with the identified code serializer to produce a class representation of the user interface object (see, for example, Barnes, column 4, lines 60-65, which shows producing a source code representation of the user interface object).

With respect to claim 46 (original), the rejection of claim 36 is incorporated, and Barnes in view of Chinnici in view of Ardoin further discloses steps for:

from the one or more serialization providers, identifying an eXtensible Markup Language (XML) serializer capable of serializing a user interface object in a serialization format that corresponds to an XML representation of the user interface object (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows identifying a serializer for serializing the user interface object in an XML representation); and

serializing the user interface object with the identified XML serializer to produce an XML representation of the user interface object (see, for example, Barnes, column 4, lines 30-35 and 44-46, which shows producing an XML representation of the user interface object).

Conclusion

5. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER